proyecto_final

January 10, 2025

1 Detección de fraudes con tarjetas de crédito

Enlace al dataset: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

1.0.1 Importa las bibliotecas necesarias

[5]:
```
# Importa la bibliotecas necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

1.0.2 Importa y organiza el dataset

[7]:
```
# Organizar los datos en un dataframe
df = pd.read_csv('creditcard.csv')
df.head()
```

[7]:

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

	V8	V9	…	V21	V22	V23	V24	V25	\
0	0.098698	0.363787	…	-0.018307	0.277838	-0.110474	0.066928	0.128539	
1	0.085102	-0.255425	…	-0.225775	-0.638672	0.101288	-0.339846	0.167170	
2	0.247676	-1.514654	…	0.247998	0.771679	0.909412	-0.689281	-0.327642	
3	0.377436	-1.387024	…	-0.108300	0.005274	-0.190321	-1.175575	0.647376	
4	-0.270533	0.817739	…	-0.009431	0.798278	-0.137458	0.141267	-0.206010	

```
        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
```

[8]:
```
df.tail()
```

[8]:
```
            Time         V1         V2        V3        V4        V5  \
284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473
284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229
284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515
284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961
284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546

                V6        V7        V8        V9  …       V21       V22  \
284802 -2.606837 -4.918215  7.305334  1.914428  …  0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  …  0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  …  0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  …  0.265245  0.800049
284806 -0.649617  1.577006 -0.414650  0.486180  …  0.261057  0.643078

             V23       V24       V25       V26       V27       V28  Amount  \
284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
284802      0
284803      0
284804      0
284805      0
284806      0

[5 rows x 31 columns]
```

[9]:
```
df.columns
```

[9]:
```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
```

```
        dtype='object')
```

[10]:
```
df.Class[df.Class == 1].count() # Transacciones fraudulentas
```

[10]:
```
492
```

[11]:
```
df.Class[df.Class == 0].count() # Transacciones no fraudulentas
```

[11]:
```
284315
```

[12]:
```
df.describe()
```

[12]:

	Time	V1	V2	V3	V4	\
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	

	V5	V6	V7	V8	V9	\
count	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	
mean	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	
std	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	
min	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	
25%	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	
50%	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	
75%	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	
max	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	

	…	V21	V22	V23	V24	\
count	…	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	
mean	…	1.654067e-16	-3.568593e-16	2.578648e-16	4.473266e-15	
std	…	7.345240e-01	7.257016e-01	6.244603e-01	6.056471e-01	
min	…	-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836627e+00	
25%	…	-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545861e-01	
50%	…	-2.945017e-02	6.781943e-03	-1.119293e-02	4.097606e-02	
75%	…	1.863772e-01	5.285536e-01	1.476421e-01	4.395266e-01	
max	…	2.720284e+01	1.050309e+01	2.252841e+01	4.584549e+00	

	V25	V26	V27	V28	Amount	\
count	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	284807.000000	
mean	5.340915e-16	1.683437e-15	-3.660091e-16	-1.227390e-16	88.349619	
std	5.212781e-01	4.822270e-01	4.036325e-01	3.300833e-01	250.120109	
min	-1.029540e+01	-2.604551e+00	-2.256568e+01	-1.543008e+01	0.000000	

```
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

Haz clic aquí para obtener una pista

```
Comienza por importar la biblioteca de pandas: import pandas as pd.

Utiliza la función read_csv() para cargar el archivo CSV en un dataframe de pandas. Especifica

Asigna al dataframe resultante al nombre de una variable, por ejemplo: data = pd.read_csv("ruta

Utiliza el método head() sobre el dataframe para mostrar las primeras 10 filas, por ejemplo: da

Asegúrate de reemplazar "ruta_al_archivo.csv" con la ruta real a tu archivo y el nombre de tu a
```

1.0.3 Limpia los datos

a. Valores perdidos

[16]:
```
df.isna().sum() # No hay valores nulos
```

[16]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
```

```
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

Haz clic aquí para obtener una pista

Utiliza el nombre de la variable del dataframe seguido del método isnull() para crear un datafi

Utiliza el método sum() en el dataframe booleano para contar la cantidad de valores verdaderos

Si unes ambos pasos, el código se verá así: data.isnull().sum()

Este código asume que el nombre del datafrme de pandas es «data». Si tu dataframe tiene un nomb

b. Datos duplicados

[19]:
```
df.duplicated().sum() # Muestra que hay 1081 filas duplicadas

df = df.drop_duplicates() # Elimina las filas duplicadas

df.duplicated().sum() # Muestra que hay 0 filas duplicadas

df.head()
```

[19]:

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

	V8	V9	…	V21	V22	V23	V24	V25	\

```
0  0.098698  0.363787  … -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  … -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  …  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  … -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  … -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
```

Haz clic aquí para obtener una pista

```
Usa el nombre de la variable del dataframe seguido del método duplicated() para crear un datafi

Usa el método sum() en el dataframe booleano para contar la cantidad de valores verdaderos (i.e

Si unes ambos pasos, el código se verá así: data.duplicated().sum()

Este código asume que el nombre del dataframe de pandas es «data». Si tu dataframe tiene un nom
```

1.0.4 Analiza los datos

Pregunta 1: ¿Cuál es el porcentaje de transacciones fraudulentas en el dataset?

[23]:
```
# Calcula el porcentaje de transacciones fraudulentas
fraud_percentage = (df['Class'].sum() / len(df)) * 100

# Muestra el porcentaje de transacciones fraudulentas
print(f"El porcentaje de transacciones fraudulentas es del {fraud_percentage:.
  ↪2f}%")
```

```
El porcentaje de transacciones fraudulentas es del 0.17%
```

Haz clic aquí para obtener una pista

Para calcular el porcentaje de transacciones fraudulentas, debes contar la cantidad de transacciones fraudulentas (aquellas donde «Class» es igual a 1) y dividirla por le número total de transacciones en el dataset. Después, multiplica el resultado por 100 para obtener el porcentaje.

Pregunta 2: ¿Cuál es el importe medio de las transacciones fraudulentas?

[26]:
```
# Calcula el importe medio de las transacciones fraudulentas
fraud_mean_amount = df[df['Class'] == 1]['Amount'].mean()

# Muestra el importe medio de las transacciones fraudulentas
```

```
print(f"El importe de las transacciones fraudulentas es de ${fraud_mean_amount:.
↪2f}")
```

```
El importe de las transacciones fraudulentas es de $123.87
```

Haz clic aquí para obtener una pista

Para calcular el importe medio de las tansacciones fraudulentas, primero deberás filtrar el dataset para que contenga solamente las transacciones fraudulentas (aquellas donde «Class» es igual a 1) y, después, calcular la media de la columna «Amount» de los datos filtrados.

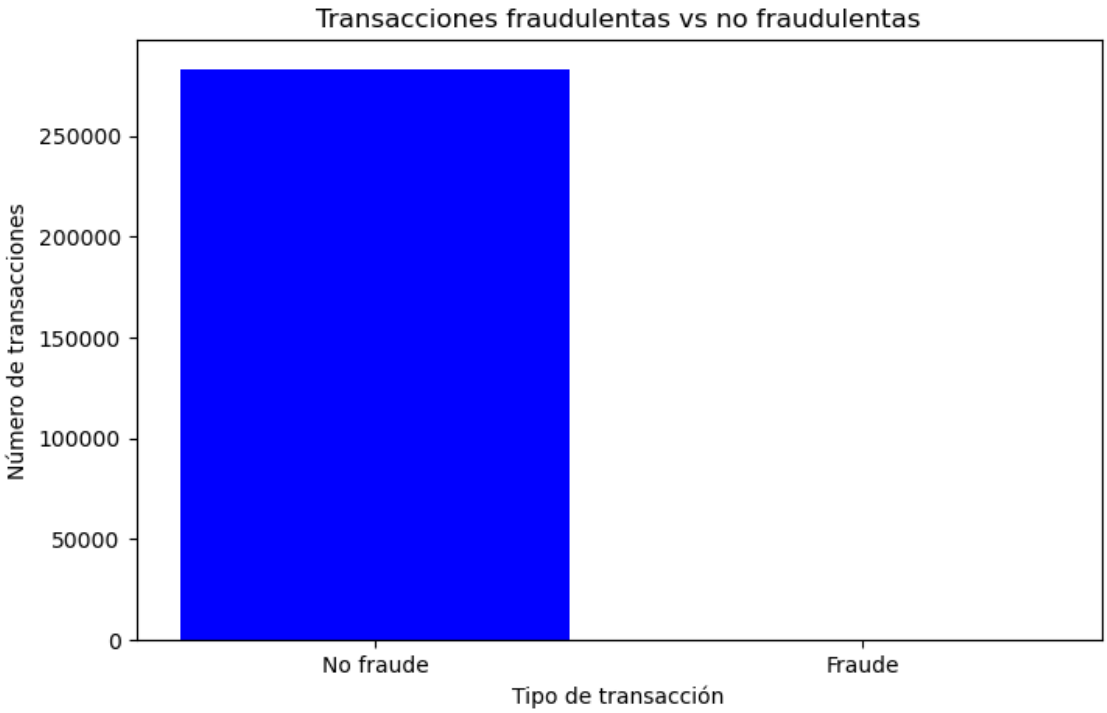
1.0.5 Visualiza los datos

Pregunta 1: ¿Cuántas transacciones fraudulentas hay en comparación con las no fraudulentas? (Utiliza un gráfico de barras)

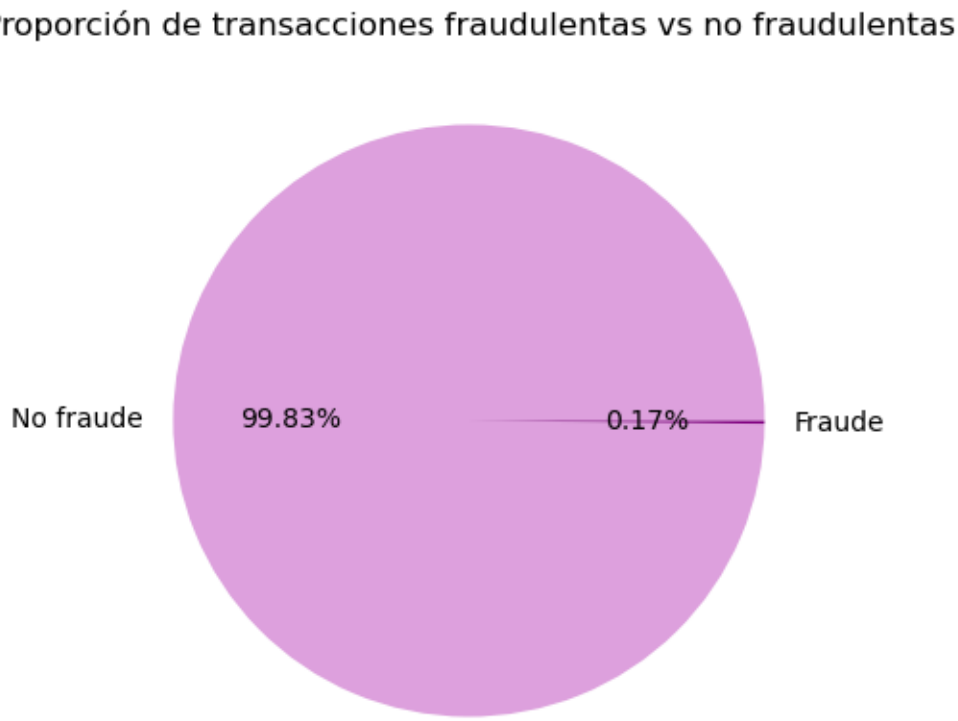
[30]:
```
# Cuenta el número de transacciones fraudulentas y no fraudulentes
fraud_counts = df['Class'].value_counts()

'''
Dado que las transacciones fraudulentas representan una fracción muy pequeña en␣
↪comparación con las no fraudulentas,
su cantidad es apenas visible en el gráfico.
'''

# Muestra la distribución de las traducciones fraudulentas con respecto de las␣
↪no fraudulentas
plt.figure(figsize=(8,5))
plt.bar(x=['No fraude', 'Fraude'], height=fraud_counts, color=['blue',␣
↪'orange'])
plt.xlabel('Tipo de transacción')
plt.ylabel('Número de transacciones')
plt.title('Transacciones fraudulentas vs no fraudulentas')
plt.show()
```



[31]:
```
# Gráfico circular para visualizar mejor la proporción de ambos tipos de␣
 ↪transacciones
plt.figure(figsize=(8,5))
plt.pie(fraud_counts, labels=['No fraude', 'Fraude'], autopct='%1.2f%%',␣
 ↪colors=['plum', 'purple'])
plt.title('Proporción de transacciones fraudulentas vs no fraudulentas')
plt.show()
```



Haz clic aquí para obtener una pista

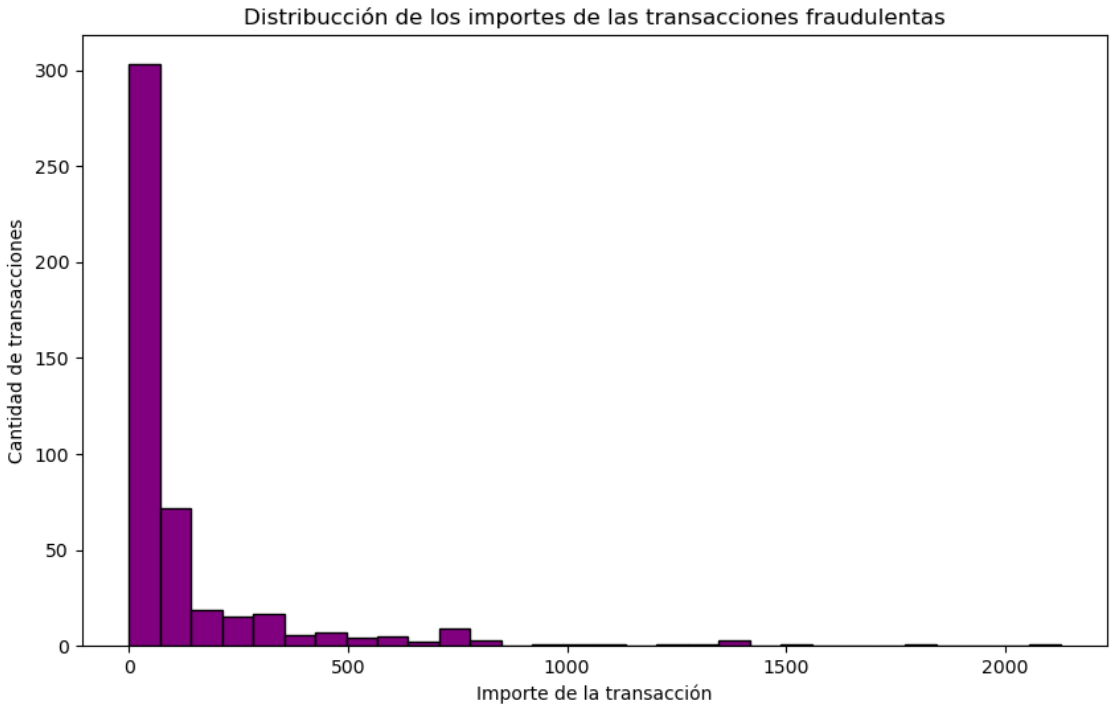
Para crear un gráfico de barras que muestre la cantidad de transacciones fraudulentas y no fraudulentas, deberás contar las veces que ocurre cada clase (fraude y no fraude) según la información de la columna «Class» y después representar estos recuentos en un gráfico de barras.

Pregunta 2: ¿Cuál es la distribución de los importes de las transacciones fraudulentas? (Utiliza un histograma)

[34]:
```
# Separa los datos de transacciones fraudulentas
fraud_data = df[df['Class'] == 1]

# Muestra la distribución de los importes de las transacciones fraudulentas
plt.figure(figsize=(10,6))
plt.hist(fraud_data['Amount'], bins=30, color='purple', edgecolor='black')
plt.xlabel('Importe de la transacción')
plt.ylabel('Cantidad de transacciones')
plt.title('Distribucción de los importes de las transacciones fraudulentas')
plt.show
```

[34]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Haz clic aquí para obtener una pista

Para visualizar la distribución de los importes de las transacciones fraudulentas, deberás filtrar el dataset para que contenga únicamente las transacciones fraudulentas (aquellas donde «Class» es igual a 1) y, después, usar un histograma para representar la distribución de los valores de la columna «Amount» de los datos filtrados.

1.1 Desarrollo y evaluación de modelos

1.1.1 Separa del dataset

[38]:
```
# Separa los datos de entrenamiento y evaluación
from sklearn.model_selection import train_test_split

X = df.drop('Class', axis=1)
y = df.Class

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[38]:
```
((226980, 30), (56746, 30), (226980,), (56746,))
```

Haz clic aquí para obtener una pista

Una vez que tegas este dataset, puedes utilizar la biblioteca scikit-learn para separar los dat

Primero, puedes crear un dataframe de pandas «X» con todas las columnas excepto la columna «Cla

A continuación, pueder usar la función train_test_split() para separar los datos en grupos de e

La función train_test_split() devuelve cuatro variables: X_train, X_test, y_train y y_test. X_t

Ten en cuenta que es importante dividir los datos en grupos de entrenamiento y de evaluación pa

1.1.2 Crea y evalúa los modelos

[41]:
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(max_depth=150, random_state=42)

model.fit(X_train, y_train)

y_predict = model.predict(X_test)
```

[42]:
```
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
import numpy as np

cm = np.array(confusion_matrix(y_test, y_predict, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['es fraudulenta', 'es normal'],␣
 ↪columns=['predicción fraudulenta', 'predicción normal'])

print("Confusion Matrix:\n", confusion)
```

```
Confusion Matrix:
                 predicción fraudulenta  predicción normal
es fraudulenta                      66                 24
es normal                            2              56654
```

[43]:
```
print("Classification Report:\n", classification_report(y_test, y_predict))#
```

```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     56656
           1       0.97      0.73      0.84        90

    accuracy                           1.00     56746
   macro avg       0.99      0.87      0.92     56746
weighted avg       1.00      1.00      1.00     56746
```

[44]:
```
accuracy = accuracy_score(y_test, y_predict) * 100
print(f"Exactitud del modelo: {accuracy:.2f}%")
```

```
Exactitud del modelo: 99.95%
```

Haz clic aquí para obtener una pista

```
Debes haber importado las bibliotecas y clases necesarias, tales como la clase RandomForestCla

Una vez hayas hecho esto, podrás crear una instancia de la clase RandomForestClassifier config

A continuación, puedes utilizar el modelo entrenado para hacer predicciones sobre los datos de

Después, puedes utilizar la función classification_report() para mostrar en la pantalla un res

Finalmente, podrás mostrar la exactitud el modelo en forma de porcentaje; utiliza el operador *
```